基于比例命中率的Web缓存区分服务

高 昂[1] 慕德俊[1] 胡延苏[1] 潘文平[2]
(西北工业大学自动化学院 西安 710072)[1] (南京航空航天大学自动化学院 南京 210016)[2]

摘 要 基于反馈控制理论,通过系统辨识设计了缓存控制器。动态调整不同类别缓存对象的缓存空间,可保证高优先级Web对象的高命中率,而不同类别的Web对象命中率之比保持不变。在服务器端实现了基于比例命中率的缓存区分服务。经实验验证,在GDSF,LRU,LFU缓存替换算法下,无论是请求命中率还是字节命中率,均有良好的区分效果。

关键词 比例命中率,Web缓存,区分服务

Differentiated Service in Web Cache Based on Proportional Hit Rate

GAO Ang[1] MU De-jun[1] HU Yan-su[1] PAN Wen-ping[2]
(College of Automation, Northwest Polytechnical University, Xi'an 710072, China)[1]
(College of Automation, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]

Abstract Based on feedback control theory, this paper designed a cache space controller through system identification. By dynamically reallocating cache space between different Web objects, the controller could guarantee the priority Web objects a higher hit rate and the proportions of hit rates between different kinds of Web objects constant. The experimental results demonstrate the proposed approach achieves the proportional hit rate guarantee under GDSF, LRU and LFU cache replacement algorithms.

Keywords Proportional hit rate, Web caching, Differentiated service

采用Web缓存技术提高站点访问速度和降低时间延迟来避免网络拥塞,已成为一个主要的研究课题。随着网络的普及,Web客户的异质性日益显现,这就要求Web服务器能够提供各种不同分类准则的区分服务[1]。本文从控制理论的角度研究基于静态Web对象的缓存区分服务。通过系统辨识,对分离式的Cache建立线性模型,进而设计控制器,使得不同类别的Web对象的命中率之比保持不变,但又能保证高优先级Web对象的高命中率,以改善用户感受。

1 背景

基于以下原因,本文对缓存中的Web对象实现基于命中率的区分服务:

1) 缓存服务器通常为众多的Web站点所共同使用,这些站点需要为其支付一定的费用,不同的站点可能对用户请求的响应时间有不同的要求。因此对于期望响应时间较小的Web站点,缓存服务器应分配更多的缓存空间以提高请求的命中率,降低响应时间。

2) 就缓存对象本身[2,3],对于包含大量内嵌对象(Embedded object)的页面,用于描述页面框架的HTML文件可能比其内部图片等的响应时间更能影响用户感受。比如在浏览门户网站时,用户可能仅需要点击HTML页面上的基于文字的超链接进而浏览感兴趣的内容,对于该页面内嵌的其它Web对象并无过多关注,或者这些对象的响应时间并不对用户后续的浏览产生重要影响。因此,HTML页面本身较内嵌对象应有更低的响应时间和更高的缓存命中率。

3) 不同的网络环境对缓存的敏感程度不同[4]。如果骨干网存在带宽瓶颈,随着Cache命中率的提高,通过骨干网访问源服务器的次数将会减少,从而减少服务时间;而如果客户端存在带宽瓶颈,即使Cache命中率提高,服务时间依然不会减少,例如对于无线标记语言(WML)描述的页面,在传输中会受到无线带宽的限制,所以缓存服务器应该避免对WML页面的缓存,而将更多的资源分配给其它的Web对象,从而达到对Cache资源的优化利用。

2 缓存区分服务

2.1 比例命中率

设L为某时段内缓存模块检测到的所有GET类型的请求总数目,对第$i(0<i\leqslant L)$个请求,$b(i)=1$表示该请求可以从缓存模块中作出响应,而$b(i)=0$则相反。设第$i(0<i\leqslant L)$个请求对应的Web对象的尺寸为$Size(i)$,那么该时间段内缓存模块的请求命中率(Request hit ratio)和字节命中率(Byte hit ratio)可分别定义为:

到稿日期:2009-05-05 返修日期:2009-07-05 本文受国防基础研究项目(C2720061361)资助。
高 昂(1984—),男,博士生,研究方向为网络QoS、网络化控制,E-mail:snailgao@gmail.com;慕德俊(1963—),男,教授,博士生导师,研究方向为计算机网络、信息安全;胡延苏(1985—),女,博士生,研究方向为网络化控制;潘文平(1980—),男,博士生,研究方向为计算机网络。

$$H_{request}=\frac{\sum_{i=1}^{L}b(i)}{L} \tag{1}$$

$$H_{byte}=\frac{\sum_{i=1}^{L}b(i)Size(i)}{\sum_{i=1}^{L}Size(i)} \tag{2}$$

本文以静态 Web 页面的缓存模块为对象，按照请求文件类型的不同，将请求的 Web 对象分为 N 个 Class，根据比例区分策略指定各个 Class 的命中率关系[5]：

$$\frac{H_{i+1}(k)}{H_i(k)}=\frac{c_{i+1}}{c_i}=Y_{i_{desire}},i=1,\cdots,N-1 \tag{3}$$

其中，H_i 代表 Class i 的请求命中率或者是字节命中率，作为 Web 缓存系统中最重要的性能指标，本文将分别验证这两种命中率情况下的缓存区分服务，c_i 代表 Class i 的固有优先级，c_i 越大，期望的命中率越高，通过动态调整提供给不同 Class 的缓存空间的大小，以实现命中率的控制。以往的研究表明，峰值命中率与缓存的大小呈指数关系[2-4]，当存在 N 类 Web 对象时，有：

$$H=k\ln\frac{S}{N} \tag{4}$$

$$H_{max}\leqslant k\ln S \tag{5}$$

其中，H 表示平均命中率，k 为平台相关的比例系数，S 为缓存空间的总大小。由式(4)和式(5)得：

$$H_{max}\leqslant\frac{H\ln S}{\ln S-\ln N} \tag{6}$$

若最高优先级与最低优先级之比 $q=c_{max}:c_{min}$，则最低优先级的 Class 的命中率上限为 H_{max}/q，以此作为确定 c_i 的依据。

2.2 分离式的 Cache 模型

图 1 为分离式的 Cache 模型，不同类的 Web 对象请求在相互分离的缓存空间被缓存[4,5]，s_i 是分配给 Class i 的缓存空间大小，$\sum_{i}^{N}s_i=S$，对于任意指定的缓存服务器，S 为有限大小。本文将反馈控制应用于分离的 Cache 模型，在缓存空间受限的情况下保证式(3)成立，控制器根据命中率观测器获得相邻 Class 命中率之比 Y_i 和期望值 $Y_{i_{desire}}$ 的残差 e_i，及调整两者缓存大小之比 $X_i=\frac{s_{i+1}}{s_i}$，从而保证不同 Web 对象命中率的比例关系。

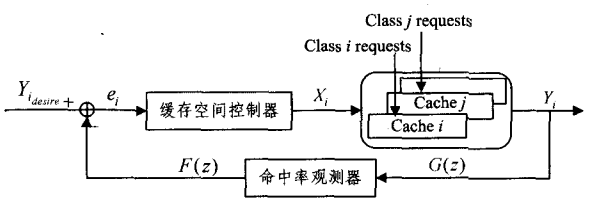


图 1 比例命中率保证的分离式 Cache 模型

3 系统建模与控制器设计

3.1 系统辨识及检验

为了设计合理的控制器，实现 Web 对象的比例命中，必须确定被控系统的数学模型，其输入输出如图 1 所示(在本文论述和实验中 $N=2$，故在下文中略去 X_i，Y_i 下角标)。虽然这种分离结构的 Cache 严格地说是非线性系统[2]，但为了简化控制器设计，本文采用线性模型近似，假定 m 阶线性模型能达到很好的近似程度，那么相应的系统差分方程可写为：

$$Y(k)=\sum_{i=1}^{m}[a_iY(k-i)+b_iX(k-i)] \tag{7}$$

对应的 Z 域为：

$$G(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{i=1}^{m}b_iz^{m-i}}{z^m-\sum_{i=1}^{m}a_iz^{m-1}} \tag{8}$$

服务器在第 k 个采样时刻，计算 Class 1 和 Class 2 的命中率之比 $Y(k)=H_2(k)/H_1(k)$，并根据伪随机序列当前的值，调整下一周期相应类别的缓存空间大小之比 $X(k+1)=s_2(k+1)/s_1(k+1)$，输入的伪随机序列按式(9)产生：

$$\varepsilon(k)=\varepsilon(k-p)+\varepsilon(k-q)\ (\mathrm{mod}\ 4) \tag{9}$$

取 $p=7$，$q=13$，具体对应关系如表 1 所列。

表 1 $\varepsilon(k)$与 $X(k+1)$对应关系

$\varepsilon(k)$	$X(k+1)$
0	1∶9
1	1∶4
2	4∶1
3	9∶1

取采样周期为 10s，实验共进行 30min，获得 150 组有效数据。采用递推最小二乘法确定系统模型参数，对于 m 阶系统，设由前 $N+m$ 次采样数据得到的参数的最小二乘估计为 θ_N，则在第 $N+m+1$ 次采样后，θ_N 可以修正为：

$$\theta_{N+1}=\theta_N+\frac{P_N\varphi_{N+1}}{\varphi_{N+1}^TP_N\varphi_{N+1}+1}[Y(N+m+1)-\varphi_{N+1}^T\theta_N] \tag{10}$$

其中，

$$P_{N+1}=P_N-\frac{P_N\varphi_{N+1}\varphi_{N+1}^TP_N}{\varphi_{N+1}^TP_N\varphi_{N+1}+1} \tag{11}$$

$$\varphi_N^T=(Y(N+m-1),\cdots Y(N),X(N+m-1),\cdots X(N)) \tag{12}$$

以上各式中 $Y(K)$与 $X(K)$均指实验获得的数据。选取适当的初值 θ_0 和 P_0，即可得到系统参数的估计值$\hat{\theta}^T=(\hat{a}_1,\cdots,\hat{a}_m,\hat{b}_1,\cdots,\hat{b}_m)$，由于估计值与真值存在误差，可定义损失函数：

$$J(m)=\sum_{k=m+1}^{m+N}\{Y(k)-\sum_{i=1}^{m}[\hat{a}_iY(k-i)+\hat{b}_iX(k-i)]\}^2 \tag{13}$$

采用 F 检验确定系统阶数。设 n_1 和 $n_2(n_1<n_2)$是模型的两个相邻阶数，构造统计量：

$$F=\frac{J(n_1)-J(n_2)}{J(n_2)}\cdot\frac{N-2n_2}{2(n_2-n_1)} \tag{14}$$

其中，N 为样本数目，$2n_1$ 和 $2n_2$ 分别对应系统阶次为 n_1 和 n_2 时系统参数的数目。当 N 充分大且 $n_2>n_1\geqslant m$ 时，F 服从分布：

$$F\sim F(2(n_2-n_1),N-2n_2) \tag{15}$$

假设 $1\leqslant m\leqslant m_{max}$，按式(10)，式(13)，式(14)分别计算 $\hat{\theta}^T$，$J(m)$以及 F 的值。给定置信度 $\alpha=5\%$，当 $m=2$ 时，$F_{0.05}(3,150)=2.66$，因为 $F_{0.05}(3,150)>F$，即当模型阶次为 2 和 3 时，$J(m)$已无显著变化，所以 $m=2$ 时，式(7)达到了足够的近似程度。因此原系统可认为是二阶系统，其中系统参数辨识如下：

$$\hat{\theta}^T=(0.6644,0.1838,0.0489,-0.0065)$$

验证结果如图 2 所示，可见，辨识值较好的近似于实际的测量值，所以用该二阶线性模型描述 Cache 分离式模型是合适的。

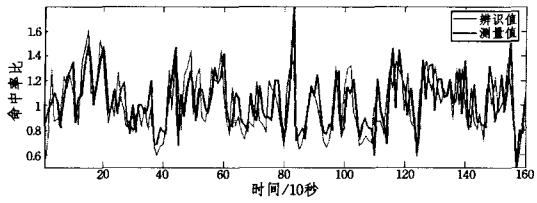


图 2　系统模型验证

3.2　控制器设计

根据图 1，系统的闭环传递函数为：

$$Y=\frac{F(z)G(z)}{1+F(z)G(z)}Y_{desire} \tag{16}$$

在理想闭环情况下，控制器应该在一个采样间隔内消除残差[2,3]，即 $Y=z^{-1}Y_{desire}$，因此有：

$$\frac{F(z)G(z)}{1+F(z)G(z)}=z^{-1} \tag{17}$$

带入 3.1 节辨识得到的系统模型(8)，得到控制器：

$$F(z)=\frac{z^{-1}}{(1-z^{-1})G(z)} \tag{18}$$

则对应的控制器差分方程为：

$$X(k)=\frac{1}{\hat{b}_1}[e(k)-\hat{a}_1e(k-1)-\hat{a}_2e(k-2)-(\hat{b}_2-\hat{b}_1)X(k-1)+\hat{b}_2X(k-2)] \tag{19}$$

4　实验验证

4.1　Test-bed

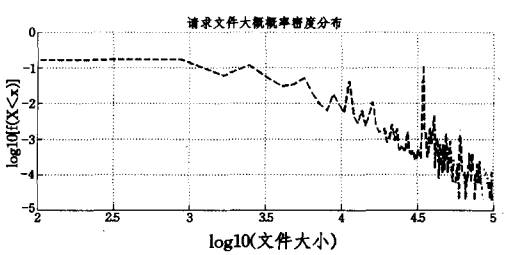


图 3　请求文件大小概率密度分布

Test-bed 由 3 台运行于 100Mbps Ethernet 的 PC 组成，所有 PC 均配置 Pentium 4 3.00GHz，512MB 内存。其中 Web Server 为运行于 Windows NT 上的 Apache 2（Httpd-ver 2.2.11），服务器端生成大小服从重尾分布的 10000 个 HTML 文件，以及 10000 个图片文件。图 3 所示为所有文件大小的概率密度分布，可见，绝大部分文件都不超过 10kbyte，并且分布具有重尾特性，符合网络通讯实际情况。配置修改过的 mod_mem_cache 模块提供基于内存的缓存，Apache 根据文件类型和基于 URL 的键在不同的缓存空间对用户请求的 Web 对象提供服务，缓存空间总大小 S 配置为 800kbyte，且保证缓存允许容纳的单个最大文件大于客户请求的最大文件。

另外两台 PC 为 Linux 系统（kernel-2.6.27），运行 SURGE（ver 1.00a）[6]作为模拟的 Web 负载，其中一台模拟 60 个客户并发请求 HTML 文件(Class 1)，另外一台模拟 60 个客户并发请求图片文件(Class 2)，设 HTML 文件具有较高的命中优先级，$Y_{desire}=7/5$，所有实验均采用 HTTP1.1 with Pipeline，单个 Client 并发请求的 TCP 连接设为 1，实验共进行 1300s，在第 500s 的时候，缓存控制器开始工作。

4.2　结果分析

图 4 所示为按式(19)设计的控制器实验结果，其中，(a)(c)(e)是以 GDFS（GreedyDual-Size）[7]，LRU（Latest-Recently-Used）和 LFU（Latest-Frequently-Used）为缓存替换算法，请求命中率 $H_i=H_{i_{request}}$ 下的实验结果，(b)(d)(f)则是相应的字节命中率 $H_i=H_{i_{byte}}$ 下的实验结果。以图 4(a)为例说明，前 500s 控制器关闭，两类 Web 对象缓存空间均为 400kbyte，同时，相应的命中率基本一致，无区分服务可言；500s 之后，在控制器的作用下，不同类别 Web 对象占用的缓存空间随之调整，对应的命中率之比也基本稳定在 Y_{desire} 附近。对比(a)(c)(e)和(b)(d)(f)还可以看出以下 5 点：

1）无论是哪种缓存替换算法，哪种命中率计算方法，缓存控制器均能够较好地实现区分服务，验证了该模型的正确性；

2）验证了式(6)，若取 $q=Y_{desire}=1.4$，则 $H_{1MAX}\leqslant 70\%$ 时，$H_{2MAX}\leqslant 50\%$，与实验结果一致；

3）比较 500s 前后（控制器开启）的缓存命中率 H_i^{-500} 和 H_i^{+500}（无论是请求命中率还是字节命中率），显然有：

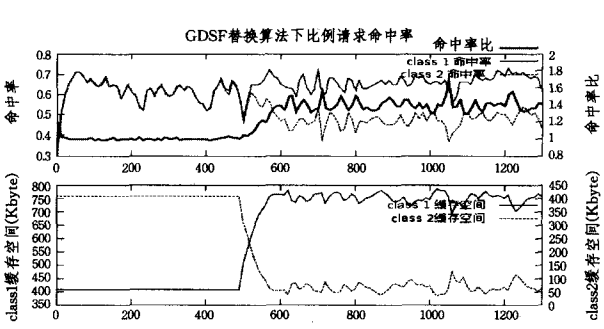
$$H_1^{+500}-H_1^{-500}\leqslant H_2^{500}-H_2^{+500} \tag{20}$$

也即，低优先级 Class 2 命中率减少的程度要大于高优先级 Class 1 命中率增加的程度，尽管如此，鉴于第 1 节的原因，实现比例命中率还是很有意义的；

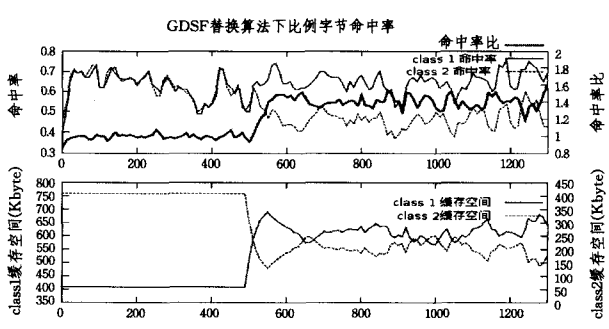
4）比较请求命中率和字节命中率，从稳定状态下（$Y=Y_{desire}$）各类 Web 对象占用的缓存空间的大小来看（无论是哪种缓存替换算法下），有：

$$\begin{aligned}&S_{require,1}^{+500}>S_{byte,1}^{+500}\\&S_{require,2}^{+500}<S_{byte,2}^{+500}\end{aligned} \tag{21}$$

其中，$S_{require,1}^{+500}$，$S_{byte,1}^{+500}$ 分别表示请求命中率和字节命中率下，稳定状态时 Class 1 的缓存空间大小，$S_{require,2}^{+500}$，$S_{byte,2}^{+500}$ 则分别表示相应情况下 Class 2 的缓存空间大小，式(21)表明，字节命中率对缓存空间比较敏感；



(a)请求命中率,GDFS



(b)字节命中率,GDFS

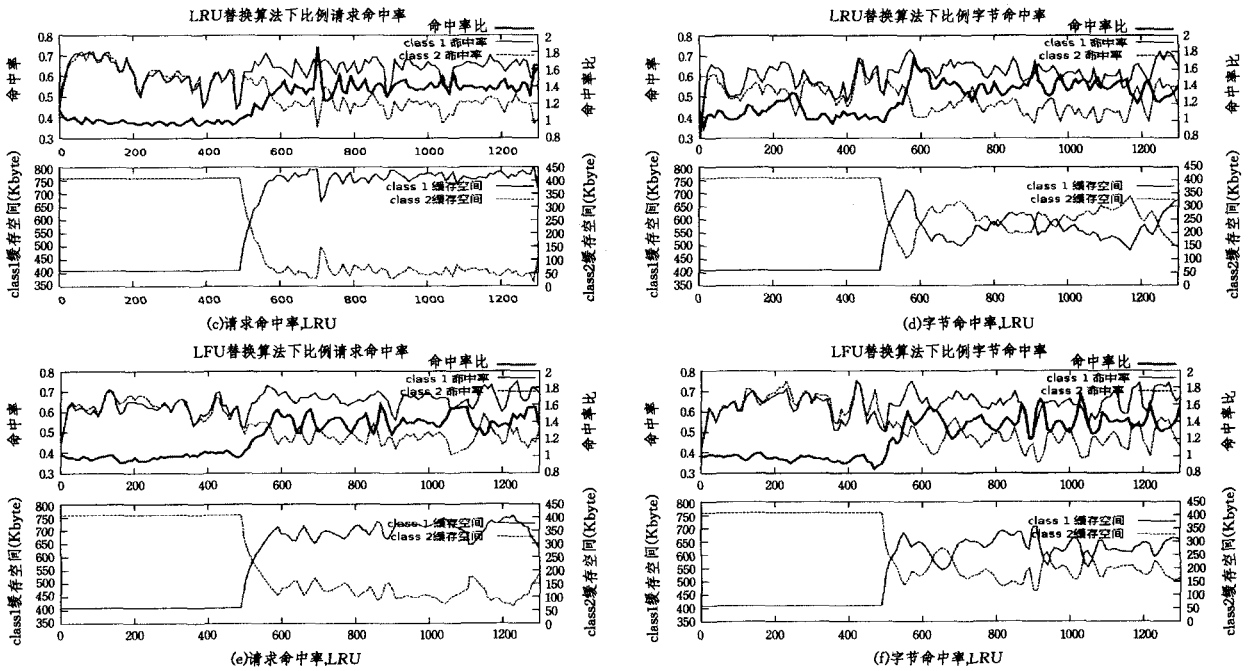


图 4 实验结果

5) 同样条件下,若以字节命中率计算,抖动较大。这是由于大文件的丢失(Miss hit)对字节命中率影响较大的缘故,而请求文件的重尾分布,使得大文件被请求的次数和频率都较小,因此无论在何种替换算法下,都不易命中,从而造成了字节命中率的抖动。

结束语 本文对静态 Web 缓存模块进行了研究,建立了比例命中率保证的分离式 Cache 结构,根据系统建模和参数辨识确定被控对象的数学模型,并在此基础上完成了缓存控制器的设计。最后,在 Windows 平台下,对 Apache2 的 mod_mem_cache 模块进行了修改并分别对 GSDF,LRU,LFU 缓存替换算法进行了实验,验证了该系统模型的有效性,实现了 Web 缓存区分服务。

参 考 文 献

[1] Dovrolis C, Stiliadis D, Ramanathan P, et al. Proportional differentiated services: delay differentiation and packet scheduling[C] //Proceedings of the ACM SIGCOMM. 1999

[2] Lu Y, Abdelzaher T F, Saxena A. Design, Implementation, and Evaluation of Differentiated Caching Services[J]. IEEE Transactions on Parallel and Distributed Systems, 2004

[3] Lu A, Saxena A, Abdelzaher T F. Differentiated caching services: a control-theoretical approach[C] // 21st International Conference on Distributed Computing Systems. 2001

[4] Venketesh P, Sivanandam S N, Manigandan S. Enhancing Qos in Web Caching using Differentiated Services[J]. International Journal of Computer Science & Applications, 2006, 3(1)

[5] Zhang Y, Feng W, Hurley R. Integration of QoS Queuing Schedules to QoS Caching Schemes[C] // Proceedings of the 5th ICISCOMSAR. 2006

[6] Barford P, Crovella M. Generating Representative Web Workloads for Network and Server Performance Evaluation[J]. ACM SIGMETRICS Performance Evaluation Review, 1998, 26(1): 151-160

[7] Cao P, Irani S. Cost-aware www proxy caching algorithms[C]// Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems. December 1997: 193-206

(上接第 35 页)

[28] Ee C T, Bajcsy R. Congestion control and fairness for many-to-one routing in sensor networks[C]// The 2nd International Conference on Embedded Networked Sensor Systems. Baltimore, MD: ACM Press, 2004: 148-161

[29] Xu Ning, Rangwala S, Chintalapudi K K, et al. A wireless sensor network for structural monitoring[C]// The 2nd International Conference on Embedded Networked Sensor Systems. Baltimore, MD: ACM Press, 2004: 13-24

[30] Donoho D L. Compressed Sensing[J]. IEEE Transactions on Information Theory, 2006, 52(4): 1289-1306

[31] Candes E J, Wakin M B. An introduction to compressive sampling[J]. IEEE Signal Processing Magazine, 2008, 25(2): 21-30

[32] Hormati A, Vetterli M. Distributed compressed sensing: sparsity models and reconstruction algorithms using annihilating filter [C]// IEEE International Conference on Acoustics, Speech and Signal Processing. Las Vegas, NV: IEEE Press, 2008: 5141-5144

[33] Haupt J, Bajwa W U, Rabbat M, et al. Compressed sensing for networked data[J]. IEEE Signal Processing Magazine, 2008, 25(2): 92-101

[34] Wan Tao, Canagarajah N, Achim A. Compressive image fusion [C]// IEEE International Conference on Acoustics, Speech, and Signal Processing. Las Vegas, Nevada: IEEE Press, 2008: 1308-1311

[35] Leung H, Chandana S, Wei Shuang. Distributed sensing based on intelligent sensor networks[J]. IEEE Circuits and Systems Magazine, 2008, 8(2): 38-52